CONSTRAINED TIME-CRITICAL ROUTING FOR MULTIPLE MOBILE AGENTS

A THESIS SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE MASTER OF SCIENCE

BY OLA FELEMBAN

DR. SHAOEN WU- ADVISOR

BALL STATE UNIVERSITY

MUNCIE, INDIANA

MAY 2017

DEDICATION

Allah and Prophet Mohammed

Whose words always encourage us in studying and achieving our goals

“My Big Family”

My mother, my father, my three brothers, one best sister, and sisters-in-law: thanks for caring, praying, and standing next to me. I could not be here without your support.

My Mother

For earning an honest living, loving, and making me confident in myself

My Father

For his patience, love, support, and care

My brothers and one sister

“My Small Sweet Family”

My husband

For his support, love, care, patience, and encourage. He does not know how much he means to me. Without him I couldn’t complete what I started.

My little daughter and son

ACKNOWLEDGEMENT

Thanks to Allah, for his countless gifts for me during my whole life.

Thanks To King Abdullah Scholarships for giving me the chance to complete my studies and taking care of studying fees.

It is a great pleasure to acknowledge my deepest thanks and gratitude to **my advisor Dr. Shaoen Wu** for his support, advice, and help. It is a great honor to work under his supervision.

Thanks to My big family: my Mother; my Father, Basim; my one and only sister Wala'a; and my sisters-in-law Hatoon, Areen, and Amaal for their big support, love, care, and prayers. **A special thanks for my brothers Emad and Muhamad** for their support and help in choosing the topic of this thesis.

I would like to express my deepest thanks and sincere appreciation for **my husband Essam Albishi** for his supporting, traveling to here with me to complete my study, standing next to me in all times, taking care of my babies to help me do my best in my study. I couldn't do it without him.

Eleen my daughter, Saif my son, you will always give me the power to complete my studies and take highest degrees. I thank Allah for blessing me with you both.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE: INTRODUCTION

In the Information Era, integrating technology with the real-world environment is a trending paradigm that attracts researchers in many fields [1]. For example, Smart Cities' applications integrate information technology with existing infrastructures, like electric grids, roads, and water pipelines, in order to optimize many aspects, such as time, energy, and cost [1][2]. In general, there is a substantial demand for the use of technology in various real-world applications that could facilitate various services provided by governmental agencies, industry, and small companies. Intelligent transportation is one of the applications that uses technologies to facilitate movements in both urban and rural road networks [2]. However, intelligent transportation solutions are prone to several challenges that limit the overall performance of vehicle routing [3]. Some of the challenges are:

- Technology deployments for intelligent transportation in smart cities
- Energy limitations of vehicles, sensors, and nodes to complete a service
- Time constraint and limitation for transportation to achieve a service in possible short time

Such challenges can be tackled by designing new methodologies that are integrated in smart cities' systems with minimal infra-structure modifications.

1.1 Research problem overview

A large number of intelligent transportation problems can be classified under Vehicle Routing Problems (VRP). This thesis focuses on solving a specific variation of VRP.

In particular, we focus on the problem where items need to be collected from different *predefined* locations while minimizing the number of participating mobile agents and the time consumed to finish the job. We assume that agents are connected to a cellular network that allows efficient communication with other agents as well as a central dispatcher.

1.2 **Research methods and goals**

In this thesis, we plan to achieve two goals that are related to each other. The first goal is to find the minimal number of agents that serve all nodes located in different locations. The second goal is to find the shortest time path of each agent. We introduce a heuristic-based solution that solves both goals simultaneously. The heuristic uses a local search algorithm to find the shortest time path for agents and a combined methodology of A* Search and Ruin and Recreate principle (R&R) to calculate the path with the shortest time for each agent.

1.3 **Thesis Preview**

The rest of this thesis is organized as follows. Chapter 2 provides a general background and related work. We explain VRP and two well-known variations of VRP. Moreover, we provide a brief introduction of graph theory and previous work that uses graph theory to represent road networks. Then, we briefly discuss prominent challenges of the time constraint in several routing and scheduling problems. Lastly, we examine methodologies used in the heuristic solution. Chapter 3 demonstrates the model and the formulation of the problem. All information required to explain the formulation is explained in detail. Chapter 4 explains the heuristic solution for the proposed problem. Chapter 5

represents the simulation results with different parameters for inputs and results in two different outputs. We provide different cases to show the different results. Finally, Chapter 6 concludes this thesis and presents future work.

CHAPTER TWO: BACKGROUND

In this chapter, we provide a formal definition of VRP and a chronological survey of various variations of VRP. In particular, we focus on the time window of VRP. Then, we present previous work on road network representations and related shortest path algorithms using graph theory. Moreover, we provide elaborate discussion about time constrained scheduling and routing. Finally, we present fundamental background for two methodologies that are prominently used for solving combinatorial optimization problems, namely Local Search and Ruin and Recreate principles.

2.1 **Vehicle Routing Problem**

VRP emerged at the end of the fifties of the last century when researchers were investigating the problem of delivering gasoline to different locations. In particular, VRP was used to find a schedule to deliver gasoline that minimized the number of trucks and overall cost [4]. VRP has a set of predefined locations and a set of vehicles, which originate from a specific location and traverse all locations in optimal routes. There exist different variations of VRP in the literature, e.g. VRP with Time Window, Capacitated Vehicle Routing, and VRP with Multiple Delivery-men [5]. In general, VRP is an NP-hard problem. Therefore, VRP and its variations are costly to solve optimally and heuristic approaches are used [6]. In the following subsection, two variants that are most related to our problem will be explained.

2.1.1 **Vehicle Routing Problem with Time Window**

The standard VRP does not take temporal consideration into account. However, in some real-world scenarios, predefined locations have time requirements to follow. In particular, the vehicles have to route to each location according to their time requirement while achieving other constraints [6]. The approach introduced in [6] is founded on the multi-objective concept, in which the goal is to accelerate the search for the original objective because that concept that was used treats a problem as it has several objectives rather than one only. VRP with temporal constraints in real-world definitely has other constraints with respect to time as the main constraint. VRP variants with temporal requirements are called Vehicle Routing Problem with Time Window (VRPTW). Similar to VRP, VRPTW belongs to the NP-hard class of problems. There are papers that extend VRPTW by considering multiple time windows, called VRPMTW. In VRPMTW, every location has multiple service time windows [7].

The above section mentioned problems do not have exact solutions to find the optimal routing solution. The authors in [8], however, introduce exact heuristic and metaheuristic methods to solve general variations of VRPTW. Two heuristics mentioned in [8] to solve VRPTW are Route-building and Route-improving. Route-building is to build new routes and Route-improving is to improve a solution from another existing solution. A metaheuristic is a combination of exploring search algorithms to improve or adjust the process of a heuristic. Exploring search concepts include Simulated annealing and Tabu search. In [9], the authors introduce an algorithm that solves a multi-objective vehicle routing problem with a time window that is based on a Genetic Algorithm called Fitness

Aggregated Genetic Algorithm (FAGA), which is basically formulated as a bi-objective model. This proposed algorithm gives better-balanced routes.

2.1.2 **Capacitated Vehicle Routing Problem**

In the capacitated VRP, vehicles have limited capacities that subsequently affect the decision of routing. The author in [10] introduces a collaborative strategy to help find the optimal solution for capacitated vehicle routing problem with a time window. This collaborative strategy is effective when the number of passengers in a station has exceeded the capacity of the servicing vehicle. The solution in such cases is to let two vehicles collaborate to deliver the customers to their destination faster, resulting in customer satisfaction.

2.2 **Graph Theory**

A graph is a structure that shows the relationship between objects [11]. Formally, a graph G is a tuple G={V, E}, where V is a set of vertices and E is a set of unweighted edges. Edges are either directed or undirected. A graph with weighted edges is a triple G={V, E, W} where W is a set of weight over edges. A graph can be connected and un-connected. A fully connected graph is a graph in which each node can be reached from any of the other nodes, whereas an un-connected graph is a graph in which at least one node can not be reached by any other nodes. A path is a trail with no repeated vertices. A trail is a walk with no repeated edges. A cycle is a closed path that starts and ends at the same vertex. A tree is a connected graph without cycles. It is common to represent a road network as a weighted graph to calculate travel time for vehicles [11]. The following

section provides discussion about an important graph problem related to road network, i.e., finding the shortest path problem.

2.2.1 **Graph Shortest Path Algorithms**

The objective of most of the recent research papers about VRP is to find the shortest path. Many graphs' algorithms are about finding the shortest path from one vertex to another with minimum cost. Table 1 shows a summary of the most popular algorithms for finding the shortest path, as well as a brief description about each one of them.

Algorithm	Description	Time Complexity	Source
Dijkstra	Find the shortest paths from a vertex to all other vertices in the graph.	$O(V^2)$	[12]
Bellman Ford	Find the shortest paths from a vertex to all other vertices in a weighted graph.	O (V.E)	[12]
A* Search	Find the optimal path between different paths for a specific goal	O(V)	[13]
Floyd warshall's	Find the shortest paths between all pairs of vertices in a graph, where each edge in the graph has a weight, which is positive or negative.	$O(V^3)$	[12]

Table 1: Famous shortest path algorithms

2.3 **Time Constrained Routing and Scheduling**

Time is a significant factor and a strong constraint that vehicles have to commit to during the routing. In fact, the number of vehicles is affected by travel time and time

windows of nodes in vehicles' schedules [14]. The authors in [14] discuss a pickup and delivery problem with a set of nodes where each has its own time window. In addition, a fleet of vehicles has a time window of its own, and all vehicles are located at a specific location called *depot*. There are many different problems discussed in [14] such as the shortest path problem with a time window (SPPTW), and the minimum spanning tree with time windows (MSTTW). SPPTW is analogous to the traveling salesman problem with a time window. Desrosiers solved this problem using a primal dual approach [14]. In [15], a problem about scheduling jobs that have time windows to a number of machines is presented. The objective is to execute the minimum number of machines to run all the jobs in their specified time windows. The authors illustrate various approximation algorithms to solve the problem. Similar to our problem, the author in [16] has a vehicle routing problem with limited capacity for vehicles that must route in minimum time and in between the time window of the scheduled nodes. The problem is called Time Dependent Vehicle Routing Problem (TDVRP). TDVRP is formed with K number of vehicles that have time windows (TW) and waits in nodes to start serving when they arrive before the node's TW.

2.4 **Heuristic Solution Methodology**

2.4.1 **Local Search method**

Local Search is a commonly used heuristic approach in finding solutions to combinatorial optimization problems [17]. Local search starts from a candidate solution and moves to a neighbor solution. According to Osman [17], there are three types of nodes

moving between two routes. Depending on the number of nodes, they move one node from one route to another, moving two nodes from a route to another, or exchanging one or two nodes between two routes. This methodology is useful for many different applications and examples in VRP, such as the Travelling Salesmen Problem and Hill Climbing [18]. Figure 1 shows the local search procedure.

Local Search Procedure:

Step 0: Specify an initial solution.

Step 1: Examine a neighborhood solution of the current solution.

Step 2: If is a better solution than [i.e.,], replace the current solution with (i.e., let) and return to Step 1.

Step 3: If all of the neighborhood solutions of the current solution have been already examined (i.e., if there is no neighborhood solution that improves) end this procedure. Otherwise, return to Step 1) [i.e., another neighborhood solution is examined in Step 1)].

Figure 1: Procedure of Local Search

2.4.2 **Ruin and Recreate method**

Ruin means mutation method and recreate means to improve the solution. The Ruin and Recreate method (R&R) has a great impact on heuristics and metaheuristics to assist to find the optimal solution. Many researchers have used this methodology due to its simplicity yet powerfulness [19]. In [19], the author introduced a hybrid algorithm of Genetic algorithm and R&R and applied it on quadratic assignment problem (QAP). In [20],

researchers said that R&R is simple and powerful for the complex problems that have many constraints, complex aims, and hard solutions. However, researchers found that the result from R&R methodology is the optimal solution. In Figure 2 below, the general procedure of R&R is presented.

Procedure R&R {ruin and recreate procedure}
  Generate (or construct) initial solution s°
  $s^{*} := recreate\ (s^{\circ})$ {$recreate\ (improve)$*the initial solution* }
  $s^{+} := s^{*}, s := s^{*}$
  repeat {main loop}
    $s := choose\_candidate\_for\_ruin(s, s^{*})$
    $s^{\sim} := ruin(s)$ {$ruin\ (reconstruct)$*the current solution* }
    $s^{*} := recreate(\sim)$ { $recreate\ (improve)$*the ruined solution*}
    if s^{*}*is better than* s^{+} ***then*** $s^{+} := s^{*}$
  until *termination criterion is satisfied*
  return s^{+}
end {*R&R*}

Figure 2: R&R procedure

CHAPTER THREE: METHODOLOGIES AND DESIGN

3.1 Network Model

We model the road network as a graph G={V, E}, where V is set of vertices (**nodes**) that are connected by a set of weighted edges E (**roads**), and the third component is tokens (**agents**) that are routed in edges.

3.1.1 NODES

There are two types of the nodes: basic and special nodes. The basic nodes NB={c0, c1,…,cn}are the **active nodes** which have items that need to be collected and require service, whereas the special nodes NS= {b0, b1} are the set of source and destination nodes that routing starts from the source and ends at the destination. The total nodes are the union of basic and special nodes. All basic nodes have two attributes: time window and load. The time window is the specific time interval that an agent should pass in between to collect items, and load is the amount of items in this node.

$\boldsymbol{t_{c_i}} = \left[\boldsymbol{t^s_{c_i}}, \boldsymbol{t^e_{c_i}}\right]$	Associated with each nodes $\boldsymbol{c_i} \in \boldsymbol{N_B}$ where $\boldsymbol{t^s_{c_i}}$ and $\boldsymbol{t^e_{c_i}}$is the start and end time of the window interval.
$\boldsymbol{W_{c_i}}$	Denotes to the load in node c_i.

Table 2: Basic node notations for time window and load

3.1.2 **Edges**

Edges represent the roads that connect nodes to each other and form a connected network. Each edge has time as a weight. The benefit of time weight is to find the optimal path with shortest time starting from the source, passing through the scheduled nodes, and ending at destination. Every token has to take the shortest path passing to all nodes scheduled to it. If the graph is a complete graph, it will not be realistic and there will be no need to make an algorithm to find the optimal path because all nodes are connected to each other. In Table 3, the notations of edges are shown.

	$\boldsymbol{T_{c_i c_j}}$ $\boldsymbol{, c_i, c_j \in N_B}$	The travel time (edge weight) from $\boldsymbol{c_i}$ to $\boldsymbol{c_j}$
$\boldsymbol{T_{b_0 c_i} = \beta}$	$\boldsymbol{, c_i \in N_B, b_0 \in N_S}$ $\boldsymbol{where\ \beta \geq \max\{T_{c_i c_j}\ i \neq j\}}$	The upfront cost from source b_0 to destination c_i.
	$\boldsymbol{T_{b_1 c_i}}$ $\boldsymbol{c_i \in N_B, b_1 \in N_S}$	The travel time from node c_i to end point b_1.

Table 3: Edge notations for travel time and upfront cost

3.1.3 **Agents**

Agents are the vehicles that pass through the nodes in the specified time window and its own time window. Agents take a trip from source to destination. It has attributes: capacity, time window, ID, and routing schedule.

Let M be the set of agents $M = \{\alpha_1, \alpha_2, \dots, \alpha_p\}$, initially located in the source, that is active to be routed. The agents have time window and capacity constraints. Notation of agents are shown in Table 4.

$\boldsymbol{t_{\alpha_k}} = [\boldsymbol{t^s_{\alpha_k}}, \boldsymbol{t^e_{\alpha_k}}]$	Is the time window interval for $\boldsymbol{\alpha_k} \in \boldsymbol{M}$ where $\boldsymbol{t^s_{\alpha_k}}$ and $\boldsymbol{t^e_{\alpha_k}}$ is the start and end time of the window interval.
$\boldsymbol{Q_{\alpha_k,}}$	Denotes to the agent α_k capacity.

Table 4: Agent notations for time window and capacity

The agent α_k must leave the source node, not before the time $t^s_{\alpha_k}$ and arrive at the destination by the time $t^e_{\alpha_k}$.

3.2 **Problem Formulation**

The following restrictions and assumptions are made to facilitate the formulations of the problem.

- Agent should operate only within its time window.
- Agents should have limited capacity.
- Nodes should be served only within the specified time window.
- Nodes should be served by exactly one agent.
- The load of each node should not exceed a specific limit.
- The total load of scheduled nodes should not exceed an agent capacity limit.

The following points are assumptions to make the formulation useful for other problems:

- All agents have the same capacity.
- The graph used for the road network is a connected component.
- Any delay in serving the nodes or late service from the vehicle is ignored.
- Service time is constant for all nodes.
- Nodes are scheduled to agents only if respective time windows match.
- The load of nodes stays constant after sending the signal to the server.
- A least one active agent is active at any time.
- Nodes are scheduled only to active agents.
- There is at least one request from a node to start the agent routing.
- Energy of agents is infinite.

3.2.1 **Variables**

The problem models a multi-agent network flow with time windows and capacity constraints. This model involves two types of variables: binary variables and continuous variables:

- **Binary Variables:**

  - $X_{c_i c_j}^{\alpha_k} = \begin{cases} 1 \; if \; the \; edge \; between \; c_i \; and \; c_j \; is \; used \; by \; agent \; \alpha_k \\ 0 \; if \; otherwise \end{cases}, \; \alpha_k \in M, c_i, c_j \in N_B$

  - $X_{b_i c_j}^{\alpha_k} = \begin{cases} 1 \; if \; the \; edge \; between \; b_f \; and \; c_j \; is \; used \; by \; agent \; \alpha_k \\ 0 \; if \; otherwise \end{cases}, \; \alpha_k \in M, \; c_j \in N_B, b_i \in N_S$

- $\zeta_{c_i}^{\alpha_k} = \begin{cases} 1 \; if \; node \; c_i \; is \; served \; by \; agent \; \alpha_k \\ 0 \; if \; otherwise \end{cases}, \alpha_k \in M, \; c_i \in N_B$ Where service at special nodes are always 1.

- $\Upsilon_{\alpha_k} = \begin{cases} 1 \; if \; agent \; \alpha_k \; is \; used, where \; \alpha_k \in M \\ 0 \; otherwise \end{cases}$

- **Continuous Variables:**
  - $S_{c_i} = service \; time \; in \; node \; c_i$, where service time is an amount of time in seconds. Notice the two special nodes $b_1 \; and \; b_0$ have no service time and S_{c_i} will be:

$$S_{b_1} = S_{b_0} = 0$$

  - $L_{c_i}^{\alpha_k}, \quad c_i \in N_B, \alpha_k \in M$ indicating a specific time at which agent α_k starts serving c_i. For instance, the agent α_1 arrived to node c_2 and start serving $L_{c_2}^{\alpha_1}$ = 5:20 pm.
  - $L_{c_j}^{\alpha_k} = \max\left\{t_{c_j}^{s}, L_{c_i}^{\alpha_k} + S_{c_i} + T_{c_i c_j}^{\alpha_k}\right\}, \quad \forall \; c_i, c_j \in N_B, \alpha_k \in M$ indicates that agent α_k starts serving c_j at the latest time: either the starting time of c_j window or arrival time of α_k. Serving time of the following node should be either the total time of routing time between the current node and the previous node; the serving time of the previous node, and the serving time of the previous node; or the starting time window of the current node, depending on which one is larger.
  - $R_{\alpha_k}, \; \alpha_k \in M$ is the routing starting time for agent α_k.

According to the above discussion, the vehicle routing and scheduling problem with time critical problems for multiple vehicles with time windows and nodes with time windows is formulated as follows:

Minimize (1)

$$\sum_{\alpha_k \in M} \sum_{c_i, cj \in N_B} (T_{c_i c_j} X_{c_i c_j}^{\alpha_k}) + \sum_{\alpha_k \in M} \sum_{c_i, cj \in N_B} (S_{c_i} \zeta_{c_i}^{\alpha_k}) + \sum_{\alpha_k \in M} \sum_{c_i, cj \in N_B} (T_{b_0 c_i} X_{b_0 c_i}^{\alpha_k} + T_{b_1 c_j} X_{b_1 c_j}^{\alpha_k})$$

The objective function (1) minimizes the total routing time cost for agents. It calculates the summation of three parts: the total time of the edges between the nodes, the time of edges from the source to the destination, and the summation of service time spent in each node.

Subject to (2)

$$\sum_{\alpha_k \in M} \sum_{c_i \in N_B} \zeta_{c_i}^{\alpha_k} = 1,$$

Equation (2) states that each basic node is visited and served by agent α_k exactly once to guarantee that no other agents can pass by it again.

$$\sum_{\alpha_k \in M} \sum_{c_j \in N_B} X_{b_0 c_j}^{\alpha_k} = 1, \qquad b_0 \in N_S \tag{3}$$

$$\sum_{\alpha_k \in M} \sum_{c_i \in N_B} X_{c_i b_1}^{\alpha_k} = 1, \quad b_1 \in N_S \tag{4}$$

Equations (3) and (4) ensure that each agent is used exactly once and that flow conservation is satisfied at each node.

$$X_{c_i c_j}^{\alpha_k} \zeta_{c_i}^{\alpha_k} \left(L_{c_i}^{\alpha_k} + S_{c_i} + T_{c_i c_j}^{\alpha_k} \right) \leq L_{c_j}^{\alpha_k}, \qquad \alpha_k \in M, (c_i, c_j) \in N_B \tag{5}$$

Equation (5) assures the consistency of the starting service time and service duration of node c_i, as well as the traveling time between c_i and c_j to start serving node c_j before its start serving time $L_{c_j}^{\alpha_k}$.

$$t_{c_i}^s \leq L_{c_i}^{\alpha_k} \leq t_{c_i}^e, \qquad \alpha_k \in M, c_i \in N_B \tag{6}$$

Equation (6) ensures that starting serving time is between the TW of the node.

$$t_{\alpha_k}^s \leq (T_{b_0 c_i} X_{b_0 c_i}^{\alpha_k}) + \sum_{c_i, cj \in N_B} (T_{c_i c_j} X_{c_i c_j}^{\alpha_k}) + \sum_{c_i, cj \in N_B} (S_{c_i} \zeta_{c_i}^{\alpha_k}) + (T_{b_1 c_j} X_{b_1 c_j}^{\alpha_k}) \leq t_{\alpha_k}^e, \quad \alpha_k \in M, \tag{7}$$

Equation (7) ensures that routing time from source to destination is within the TW of the agent.

$$\sum_{c_i \in N_B} W_{c_i} \zeta_{c_i}^{\alpha_k} \leq Q_{\alpha_k}, \quad \alpha_k \in M \tag{8}$$

Equation (8) is for the capacity constraint of the agent.

$$t_{\alpha_k}^s \leq R_{\alpha_k} \leq t_{\alpha_k}^e \tag{9}$$

Equation (9) ensures that routing starting time of agent is within its time window.

CHAPTER FOUR: HUERISTIC APPROACH

Finding an optimal solution of VRP problems is infeasible [6]. Most research papers, such as [10], [21], [22], [23], and [24] find solutions that are suboptimal called Heuristic solutions [24]. Our heuristic approach is developed to minimize the number of agents that achieve minimum traveling time after serving all nodes. The following points summarize the general idea of the proposed heuristic approach.

- The more agents that are used, the greater the cost, hence the more total time for all agents.
- The more agents that are used, the faster each node will be helped, hence the less time spent on roads per agent.

This heuristic approach is composed of two stages: scheduling nodes to agents and finding the shortest time path for agents. In brief, this heuristic approach starts with an equal number of nodes and agents. Each node is nominated to each agent using the Insertion Construction Algorithm. After that, the shortest time paths will be calculated for each agent using a combination of A* Search and R&R algorithms. Finally, the node that has longest time path will be transferred into another agent that is closer to it using the local search method. This process is repeated until all nodes are served. The heuristic in high-level language is described in Figure 3 below.

1. Let *N* be number of basic nodes and agents, RT array is the routing times array for all agents of size *N*. Each agent will have an array, *TT* integer variable for total routing time, *TempTT* integer variable for temporary total time, *N* number of *AgentSchedule* array is agents' schedule. The array's size will be equal to the capacity *Q* of the agents. *TempLargeTime* variable is the largest routing time from the array *RT,* counter *i.*

2. Set road network graph (nodes and edges) and place agents in the source node.

3. Set the edges' weights for every edge connected to source b_0 and set source and destination.

4. Do Insertion Construction Algorithm (described in subsection 4.1).

5. For each agent, iteration starts:

5.1. Set *i*=1, a counter for number of agents' arrays that has i nodes, the i increment when *TempLargeTime* hold the last large routing time of agent that has i nodes.

5.2. Calculate each agent's shortest routing time by using A* search and R&R algorithm (described in 4.3) and save it in *RT*. save the largest value of *RT* in *TempLargeTime.*

5.3. Calculate the summation of all routes' times of array 2 and save it in *TT (*and *TempTT* for the first iteration*)*.

5.4. Use Local Search Method (described in 4.2).

5.4.1. Move node/s that has the value in *TempLargeTime*, name it *S,* to the agent's schedule that has the next lexicographic neighbor node, or the next lexicographic un-neighbor node and name it $\grave{S}$, and the capacity of that agent is enough to move the node/s into its schedule.

5.4.2. Update the $\grave{S}$ schedule.

5.4.3. Delete *S* the old agent schedule.

5.4.4. Decrease the number of agents *N.*

5.5. Compare *TT* and *TempTT* and add the difference into *variance* array. Add *TT* into *TempTT* if this is the second iteration.

Figure 3: A high-level description of the heuristic approach

4.1 **Insertion Construction Algorithm**

The main feature of the insertion algorithm is to generate a general schedule for all basic nodes to all agents. For instance, the solution generates an equal number of nodes and agents to start. The heuristic will start with generating a schedule for each agent and assign a node to each agent. Thus, each agent will have a node in its schedule. When all nodes are scheduled, then the construction ends.

4.2 **Local Search Method**

Local Search (LS) method is used in this heuristic approach to minimize the number of routes and decrease the total travel time by reducing the number of active agents. In our heuristic solution, moving nodes from one schedule to another depends on a number of operators: $N\alpha_k$ the number of nodes in an agent's schedule, Q_{α_k} the capacity of an agent α_k, and i, which is a counter. The basic idea of Local Search is that the node that has the largest routing time can be transferred into the next lexicographic neighbor or next lexicographic un-neighbor if the capacity of that neighbor's agent is enough to have it in its schedule. We repeat the algorithm until all nodes are served with a minimum number of agents and minimal traveling time. The explanation of Local Search Method in high-level language appears in Figure 4.

1. Move node/s that has the largest tour time, name it S, to the agent's schedule that has the next lexicographic neighbor node, or the next lexicographic un-neighbor node and name it S, and the capacity of that agent is enough to move the node/s into it's schedule.
2. Update the S schedule.
3. Delete S the old agent schedule.
4. Decrease the number of agents N

Figure 4: A high-level description of the Local Search Method

4.3 **A* Search and R&R Algorithms**

A* Search algorithm is used to find the shortest path from one point to the related points in a road network graph. A* Search is combined with the Ruin and Recreate principle to get the best results. In simple words, the Ruin and Recreate principle is suited for complex problems to help in making heuristics for optimal solutions. Combining these two methodologies help in achieving the heuristic goal, which is finding the optimal shortest time path for all available agents.

This heuristic is found and built for our problem after creating the algorithm from scratch manually and solving a small example by hand. The handwritten solution helps in finding the heuristic algorithm to get the optimal solution for the problem. In addition, the handwritten solution helps to track the function of minimizing the number of agents. Moreover, it helps to see when the iterations will stop and get the result when having the perfect number of agents with groups of nodes. As shown in Figure 5, two lines present the simulation results and handwritten results of 10 nodes and 10 agents as a starting number. The convergence of the two lines is proof to show the results are optimal for the problem.

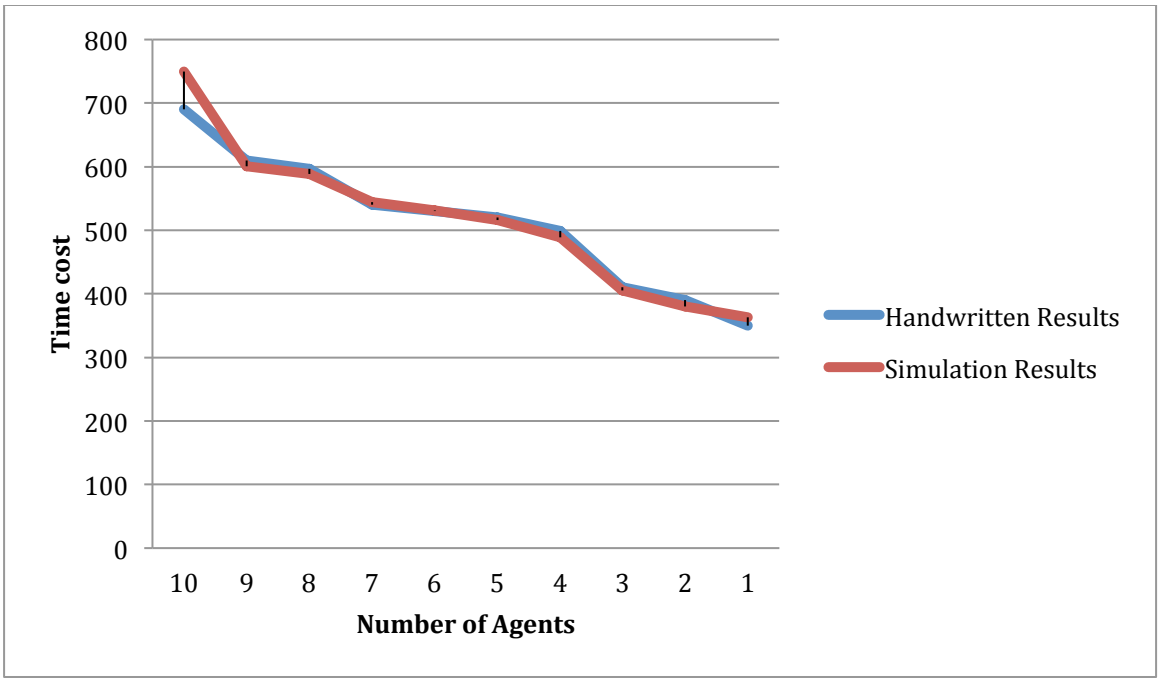

Figure 5: A comparison chart between the simulation solution and the handwritten solution of 10 nodes and agents

CHAPTER FIVE: SIMULATION RESULTS

To implement the heuristic solution, a java-based open-source library called Jspirit is used. Jsprit is a library developed to solve many variants of Vehicle Routing Problems [25]. Jsprit accepts input in two ways: either hard coding in java language or by an XML file [25]. Either way, the user writes all needed information for nodes, as shown in Table 5, and for agents, as shown in Table 6.

A service information	
Id	"46"
Type	"Delivery"
LocationId	X=30, Y=32
Coordination	X= 30, Y=32
Capacity-demand	30
Duration	90.0
Time window	Start=448, end=509

Table 5: An example of a service element from XML file

A vehicle information	
ID	Vehicle
Type ID	HVRPType
Start location	Id=0, Coordination x="40.0", y="50.0"
End location	Id=101, Coordination x="70.0", y="10.0"
Time schedule	Start =0.0, end=1236.0
Turn to depot	True
Vehicle Type information	
ID	HVRPType
Capacity	250

Table 6: An example of a vehicle element from an XML file

Each service, i.e. node, in Jsprit has an Identification number (ID), a service type, LocationId, Coordination, Capacity-demand, Duration, and Time window. The service type is one of three types: pickup, delivery, or shipment. LocationId is the identification of the service's location to guarantee that no other service is located in the same location. Coordination is the location of the service in the graph plot, which is generated randomly. Capacity-demand is the load that each service needs to be served. Duration is the amount of time a service is being served. Finally, a time window is period of time that has a starting and ending time. Each service should have all the information required to complete the program input.

Unlike services, vehicles' information are not required to be repeated for all vehicles used in the program because we limit the vehicles to be all the same type and size. It is only written one time and the code will generate or remove vehicles as needed from the code. A vehicle has *ID, Type ID, Start Location, End Location, Time schedule, and Turn to depot. Start location* and is where the vehicles are originally located and start routing from; *End location* is where the vehicles' route ends. *Time schedule* has a start and end time; the end time is when the vehicle stops serving at and goes back to the depot. *Turn to depot* is a Boolean variable. Two important data in vehicle type information are ID and capacity, as shown in Table 6. Capacity is the limitation of load for all agents.

After running the code, the number of nodes shows in the console window. An example of typical output including indicators and their values is shown in Table 7. Indicators are: *noJobs* and *noServices* are the number of nodes that are given by the user. *noShipments* is the number of shipments, which indicates the quantity of loads that needs pickup and delivery from a node. *noBreaks* is the number of times that each agent has to

pause routing for any condition. *Fleetsize* is the number of agents before starting the problem. Fleetsize is given as infinity for the initial value; then it will take the value of number of nodes.

The output can be in either a graph plot or tables of results in the console window. A graph plot displays the basic nodes as blue points, special nodes as red nodes and vehicle's routing as colored lines between nodes. The table of results, as shown in Table 8, has indicators and its values. Indicators are: *cost, noVehicles, unassgndJobs, time cost without waiting, and time cost with waiting*. *Cost* is the total distance of all agents' routings start at the source and end at the destination. *noVehicle* is the final number of vehicles after reducing the number of routes. *unassgndJobs* is the number of unassigned jobs or services to any vehicle. *Time cost without waiting* is the focus in this thesis, which it is the time spent in serving nodes without the wasted time before the node's starting time window. *Cost with waiting time* is the opposite. In the following subsection, the experiment settings of the heuristic solution are described in detail.

5.1 **Experiment settings**

In the program created to match our heuristic approach, the programmer inserts the input in an XML file because it is easier and more flexible when editing, adding, and deleting services. The results appear in tables in the console window and in graph plots. Some of the information of services and agents shown in Table 5 and 6 respectively are adjustable and can be used in different experiments.

In this thesis, we conduct two experiments to monitor the time and cost spent during vehicles' routes by changing two variables: the number of nodes and the nodes' duration. In each experiment, we fix one variable and change the other. All settings'

variables are then randomly added by the user. In the following section, two case studies will be presented to show the time cost changes in changing the number of nodes and duration variables.

5.2 **Case Studies**

In this section, two case studies are presented: the changing number of nodes case, and the changing duration of nodes case.

5.2.1 **Changing number of nodes**

In the first case, the number of nodes is changed where all other variables are fixed during the study. Four experiments have been implemented with four different numbers of nodes to show the traveling time cost changes; 100, 50, 25, and 10, respectively. In the first experiment, the number of nodes is set to 100 basic nodes located at random unique locations in the graph, and 2 special for the source and destination nodes. Thus, 100 agents will be created and located in the source node. The input is shown in Table 7, and the output is shown in Table 8. As shown in Table 8, the number of vehicles is minimized when traveling time cost is also decreased. In Figure 6, a chart shows the time cost during the reduction of the number of agents in some chosen iterations in the first experiment. The final result, 14 agents in 3571.0 seconds, is the optimal solution because iterations cannot find any better, which it is the last point in the line shown in Figure 6. As shown in Figure 7, the x-axis is the number of nodes and the y-axis is traveling time cost. it appears that the line goes up when the number of nodes increases, as well as number of agents. Obviously, when the

number the nodes increases, more agents are needed to serve all nodes in a short time. Additionally, more time is spent by agents to serve all nodes. The times presented in Table 8 are the optimal traveling time cost after decreasing number of used agents. Figures 8, 9, 10 and 11 are the graph plots of the four experiments respectively.

Indicator	Experiment #1 Value	Experiment #2 Value	Experiment #3 Value	Experiment #4 Value
NoJobs	100	50	25	10
NoServices	100	50	25	10
NoShipments	0	0	0	0
NoBreaks	0	0	0	0
Fleetsize	INFINITE	INFINITE	INFINITE	INFINITE

Table 7: Problem inputs of the four experiments

Indicator	Experiment #1 Value	Experiment #2 Value	Experiment #3 Value	Experiment #4 Value
Costs	14000.0	7000.0	3000.0	2000.0
NoVehicles	14	7	3	2
UnassignedJobs	0	0	0	0
Time cost without waiting	3571.0	1681.0	640.0	363.0
Time cost with waiting	13418.0	6983.0	3539.0	2235.0

Table 8: Final solution for the four experiments

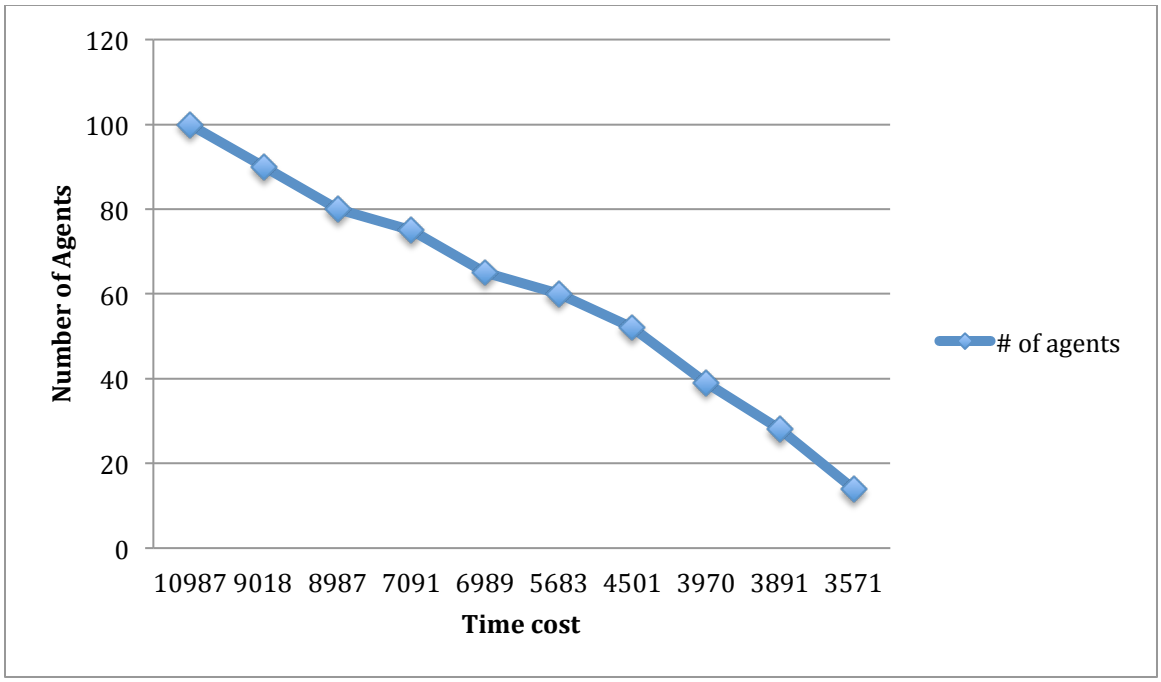

Figure 6: Time cost during number of agents decreasing

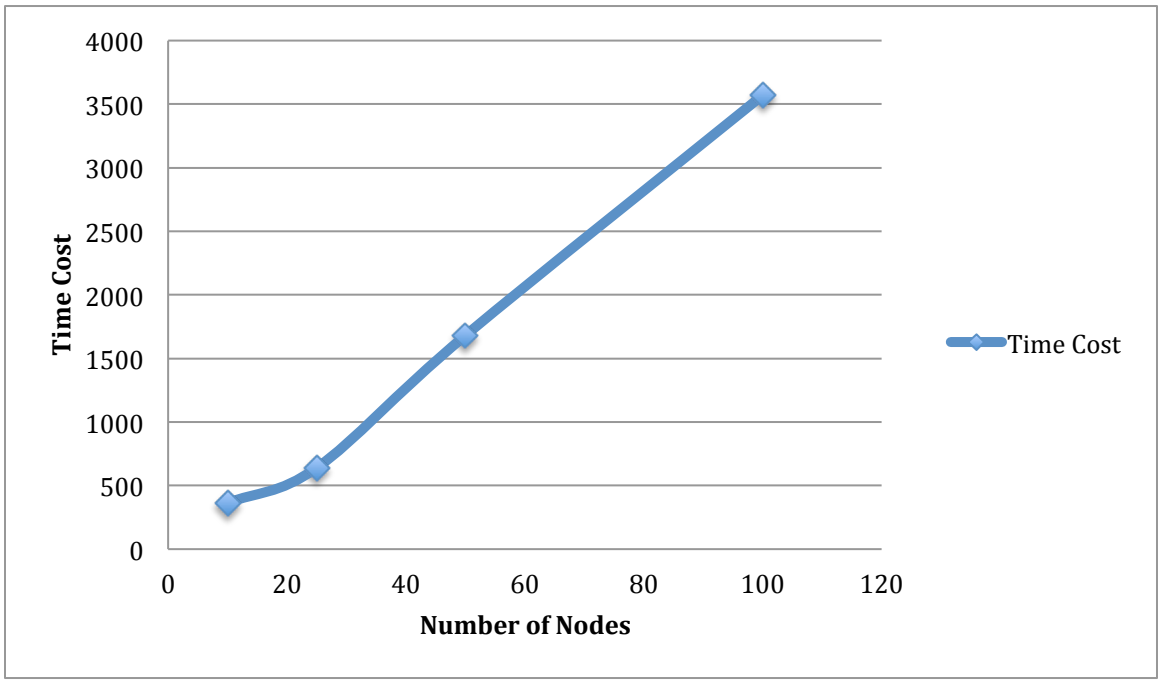

Figure 7: Time cost after changing the number of nodes

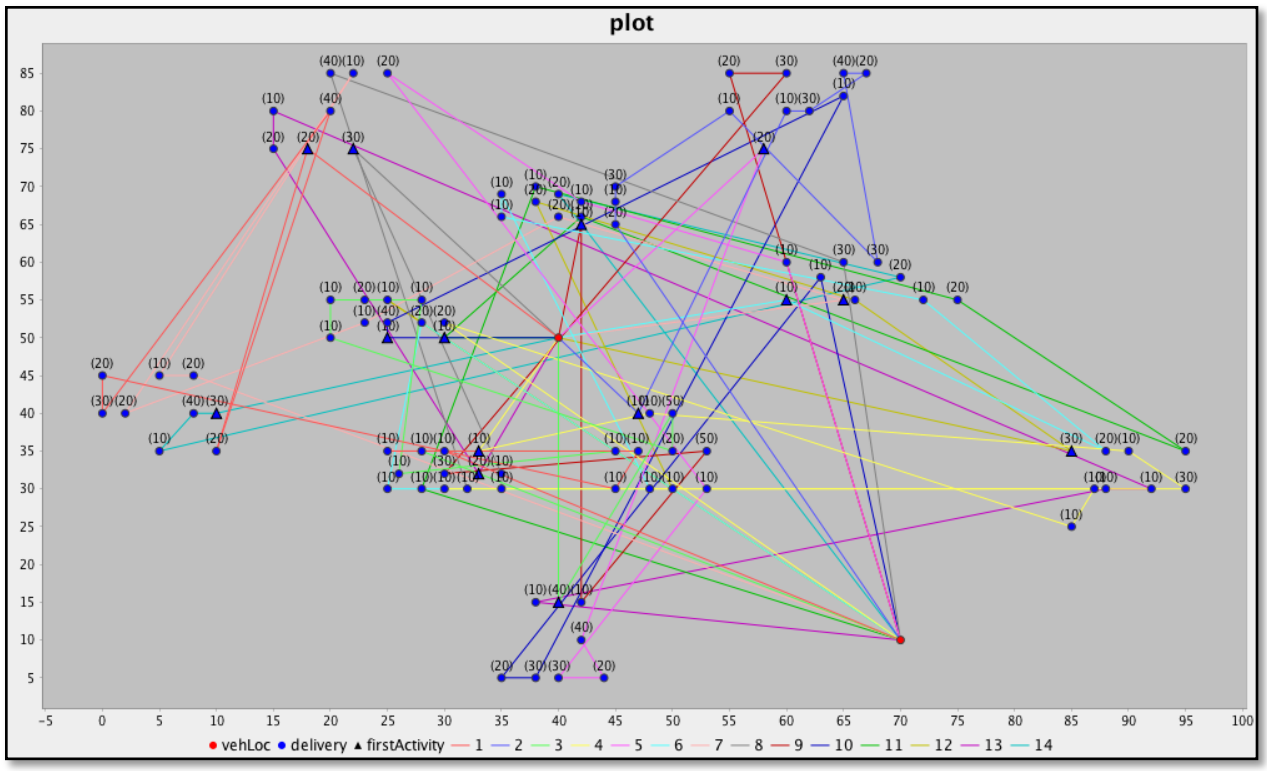

Figure 8: A graph plot of 100 nodes and 14 agents

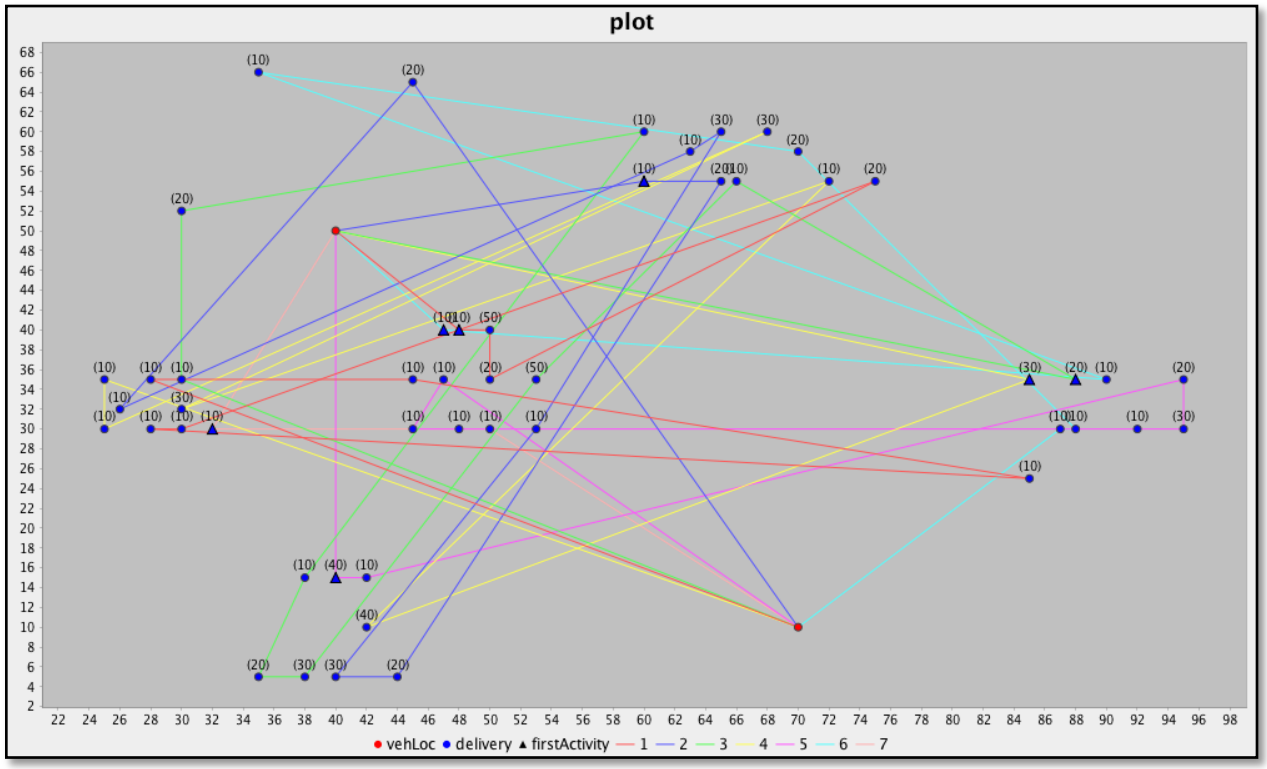

Figure 9: A graph plot of 50 nodes and 7 agents

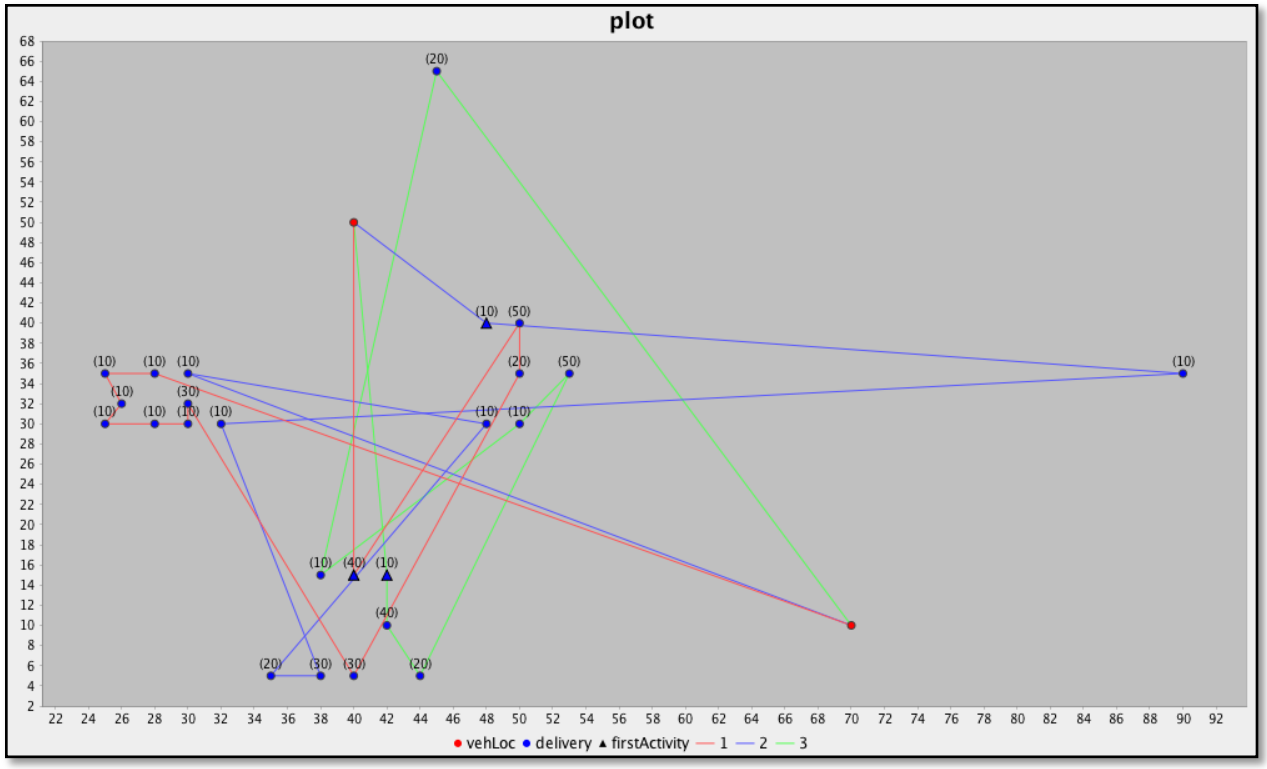

Figure 10: A graph plot of 25 nodes and 3 agents

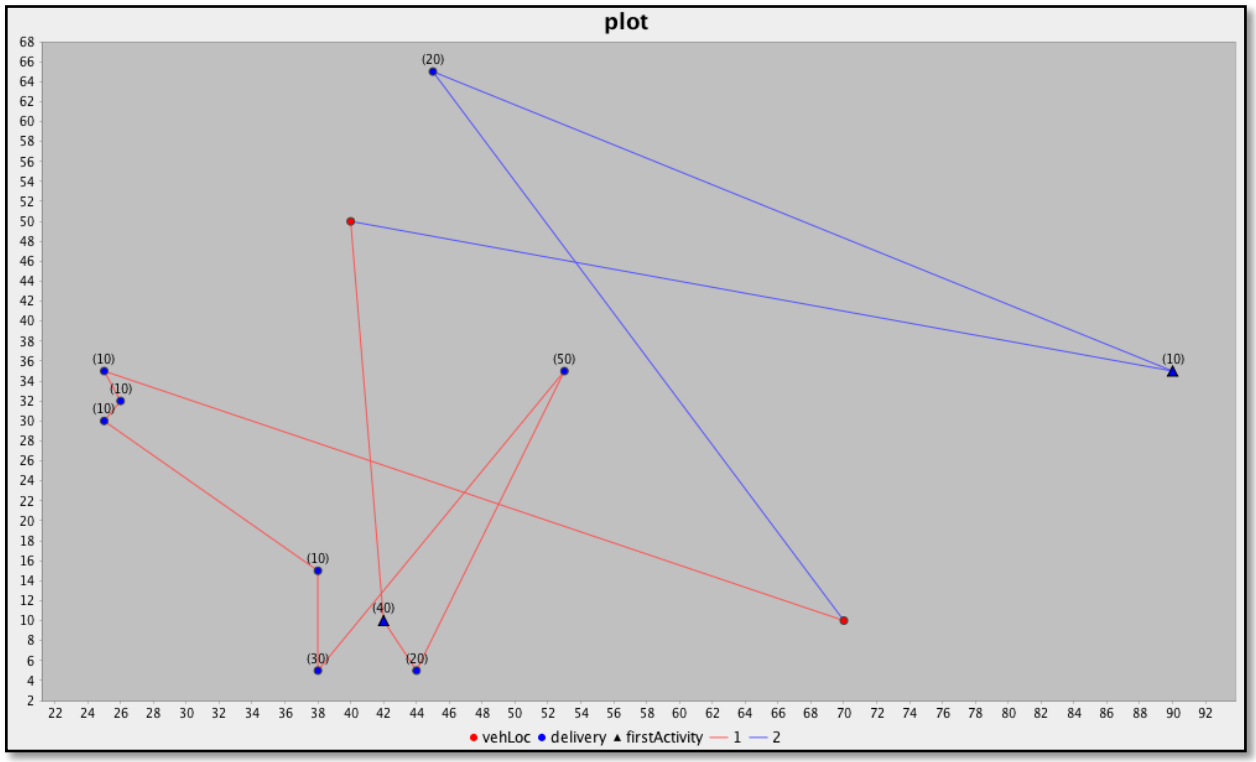

Figure 11: A graph plot of 10 nodes and 2 agents

In the next subsection, the variable is duration of nodes to monitor the time cost changes.

5.2.2 **Changing Nodes' Duration**

In this subsection, 4 experiments have been applied with 10 nodes and 4 different nodes' service durations: 50.0 seconds, 90.0 seconds, 150.0 seconds, and 250.0 seconds. As explained previously, *Duration* is the length of time spent by an agent to carry the load. The number of nodes in all experiments is constant. The first experiment starts with the shortest duration, 50.0 seconds. Table 9 shows detailed information about vehicle routes: *Route* is number of routes that have been done, *Vehicle* is the name of the vehicles used in in this routing, *Activity* is the type of service, *Job* is the service ID, *ArrTime* and *EndTime* are arriving and leaving times. It also shows the service starting and ending time. *Costs* are the distance value between two nodes' points. Finally, *Time costs* represents the time spent in serving nodes.

Detailed Solution							
Route	Vehicle	Activity	Job	ArrTime	EndTime	Costs	Time costs
1	Vehicle	Start	Depot	Undef	0	0	0
1	Vehicle	Delivery	43	17	67	0	17.0
1	Vehicle	Delivery	42	70	120	0	3.0
1	Vehicle	Delivery	33	146	196	0	26.0
1	Vehicle	Delivery	41	224	274	0	28.0
1	Vehicle	Delivery	35	304	354	0	30.0
1	Vehicle	Delivery	37	360	433	0	6.0
1	Vehicle	Delivery	38	435	529	0	2.0
1	Vehicle	Delivery	39	534	617	0	5.0
1	Vehicle	Delivery	36	622	715	0	5.0
1	Vehicle	Delivery	34	718	801	0	3.0
1	Vehicle	end	Destination	872	undef	0	71.0

Table 9: A detailed solution of a 50.0-second duration

Four experiments' inputs are shown in Table 10. As mentioned before, 10 nodes is a fixed number in all experiments of this case study to show that duration changing makes a difference in time cost. In Table 11, the solution values are shown. Notice the change in time costs without waiting and the number of vehicles, as noted previously, when the number of agents increases. However, increasing the number of agents here is not related to the number of nodes. It is related to the agents' time windows and the length of service durations in nodes. This is illustrated in Figure 12, where the x-axis is the nodes' duration and the y-axis is the time cost. The line between them goes up when both values increase.

Indicator	Experiment #1 Value	Experiment #2 Value	Experiment #3 Value	Experiment #4 Value
Number of Nodes	10	10	10	10
Nodes' Duration	50	90	150	250

Table 10: The problem input of the four experiments

Indicator	Experiment #1 Value	Experiment #2 Value	Experiment #3 Value	Experiment #4 Value
Costs	1000.0	2000.0	3000.0	4000.0
NoVehicles	1	2	3	4
UnassignedJobs	0	0	0	0
Time cost without waiting	196.0	247.0	375.0	504.0
Time cost with waiting	872.0	1200.0	2167.0	3618.0

Table 11: The result of the four experiments

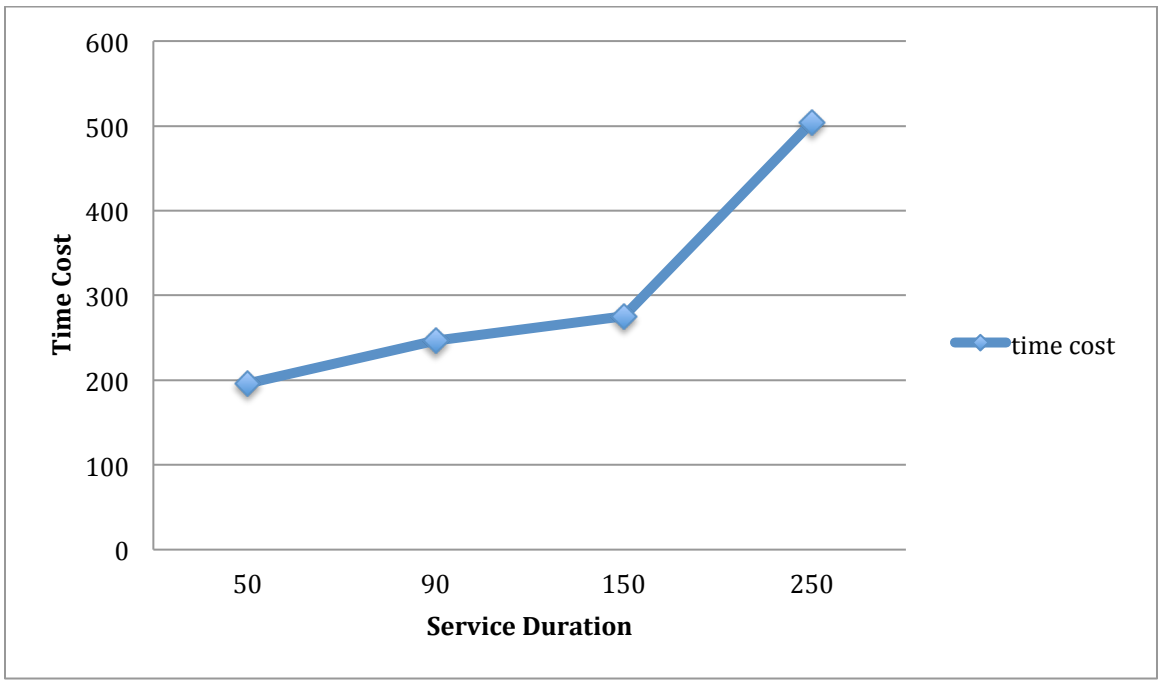

Figure 12: Time cost after changing nodes' duration

In Figures 13, 14, 15 and 16 are the graph plots of the four experiments when increasing the duration time from 50.0 to 90.0, 150.0, and 250.0, respectively. In addition, Tables 9, 12, 13, and 14 show the detailed solution of the four experiments.

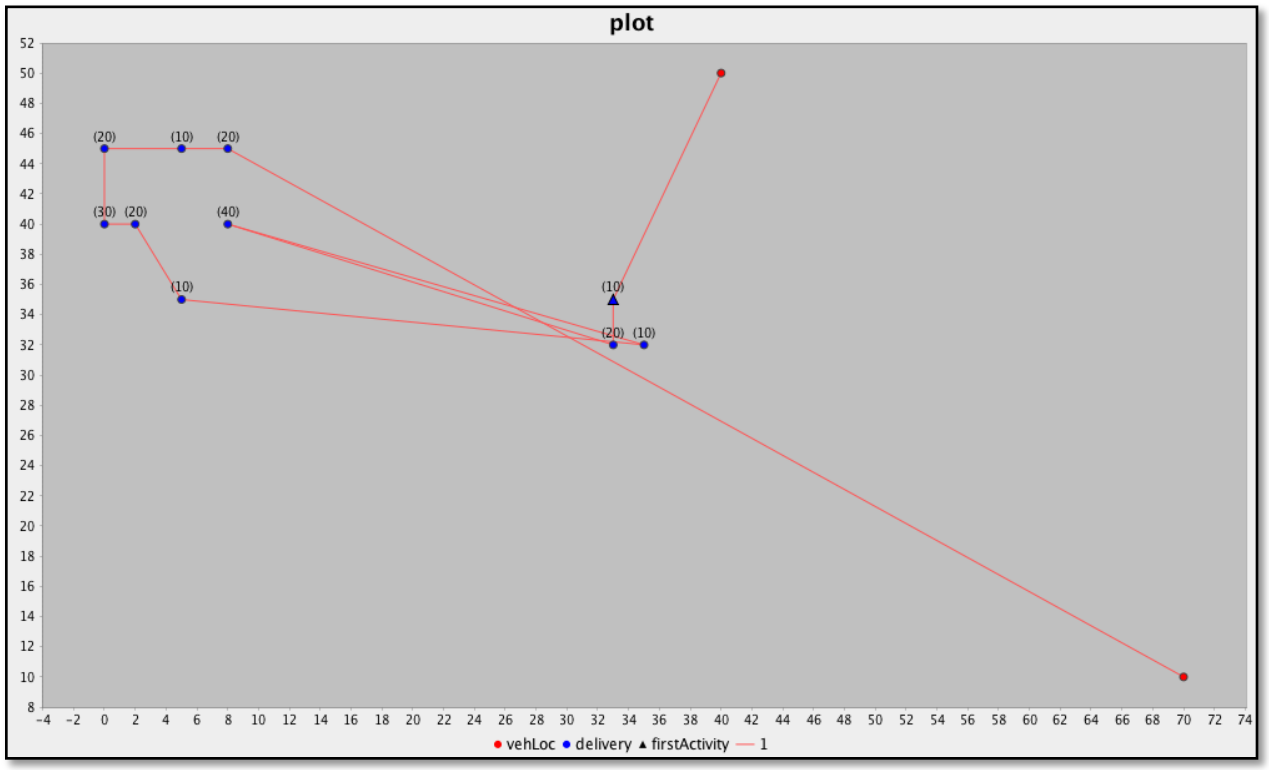

Figure 13: A graph plot of nodes with a 50.0-sec. duration

Detailed Solution							
Route	Vehicle	Activity	Job	ArrTime	EndTime	Costs	Time costs
1	Vehicle	Start	Depot	Undef	0	0	0
1	Vehicle	Delivery	43	17	107	0	17.0
1	Vehicle	Delivery	42	110	200	0	3.0
1	Vehicle	Delivery	41	202	292	0	2.0
1	Vehicle	Delivery	35	322	412	0	30.0
1	Vehicle	Delivery	37	418	508	0	6.0
1	Vehicle	Delivery	38	510	600	0	2.0
1	Vehicle	Delivery	39	605	695	0	5.0
1	Vehicle	Delivery	36	700	790	0	5.0
1	Vehicle	Delivery	34	793	883	0	3.0
1	Vehicle	End	Destination	954	Undef	0	71.0
2	Vehicle	Start	Depot	Undef	0	0	0
2	Vehicle	Delivery	33	34	177	0	34.0
2	Vehicle	End	Destination	246	Undef	0	69.0

Table 12: The detailed solution of a 90.0-sec. duration of nodes

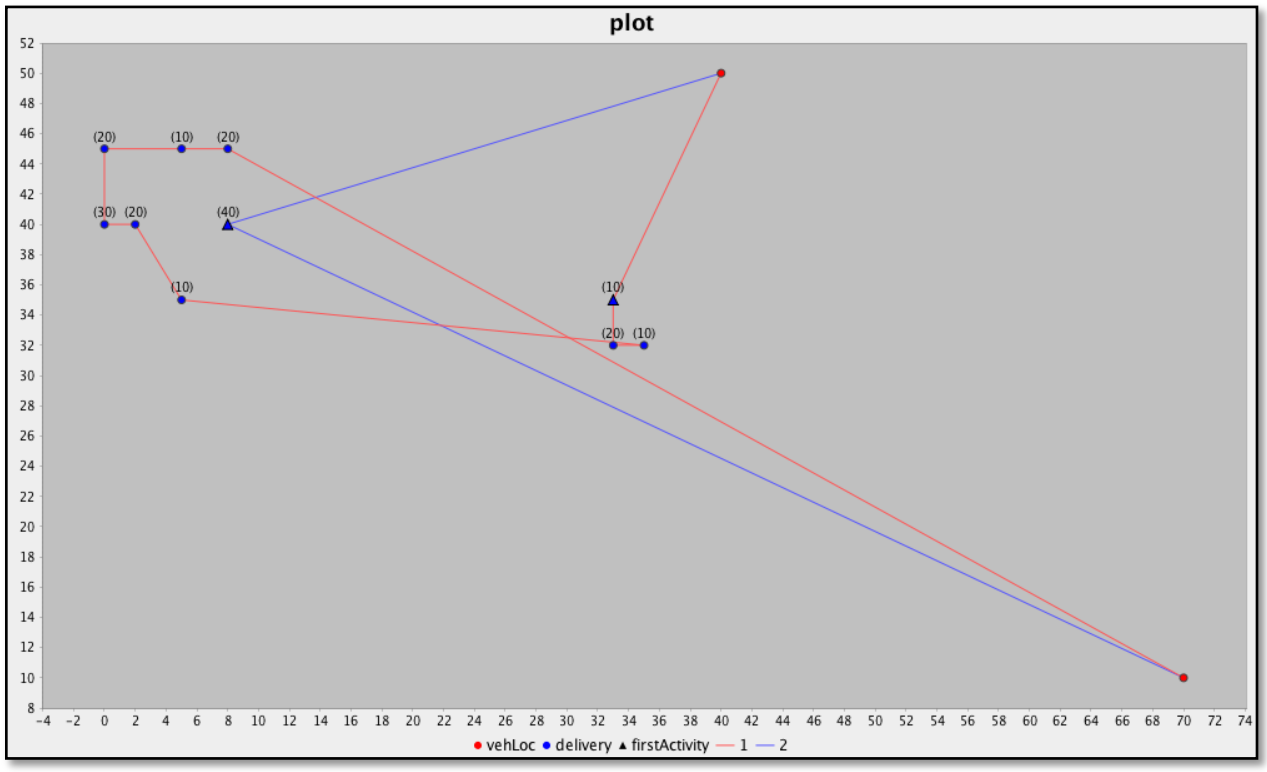

Figure 14: A graph plot of nodes with a 90.0-sec. duration

Detailed Solution							
Route	Vehicle	Activity	Job	ArrTime	EndTime	Costs	Time costs
1	Vehicle	Start	Depot	Undef	0	0	0
1	Vehicle	Delivery	43	17	167	0	17.0
1	Vehicle	Delivery	35	195	433	0	28.0
1	Vehicle	Delivery	38	440	629	0	7.0
1	Vehicle	Delivery	36	636	815	0	7.0
1	Vehicle	End	Destination	889	Undef	0	74.0
2	Vehicle	Start	Depot	Undef	0	0	0
2	Vehicle	Delivery	42	19	218	0	19.0
2	Vehicle	Delivery	41	220	370	0	2.0
2	Vehicle	Delivery	37	404	554	0	34.0
2	Vehicle	Delivery	39	559	717	0	5.0
2	Vehicle	Delivery	34	725	901	0	8.0
2	Vehicle	End	Destination	972	Undef	0	71.0
3	Vehicle	Start	Depot	Undef	0	0	0
3	Vehicle	Delivery	33	34	237	0	34.0
3	Vehicle	End	Destination	306	Undef	0	69.0

Table 13: A detailed solution of a 150.0-sec. duration

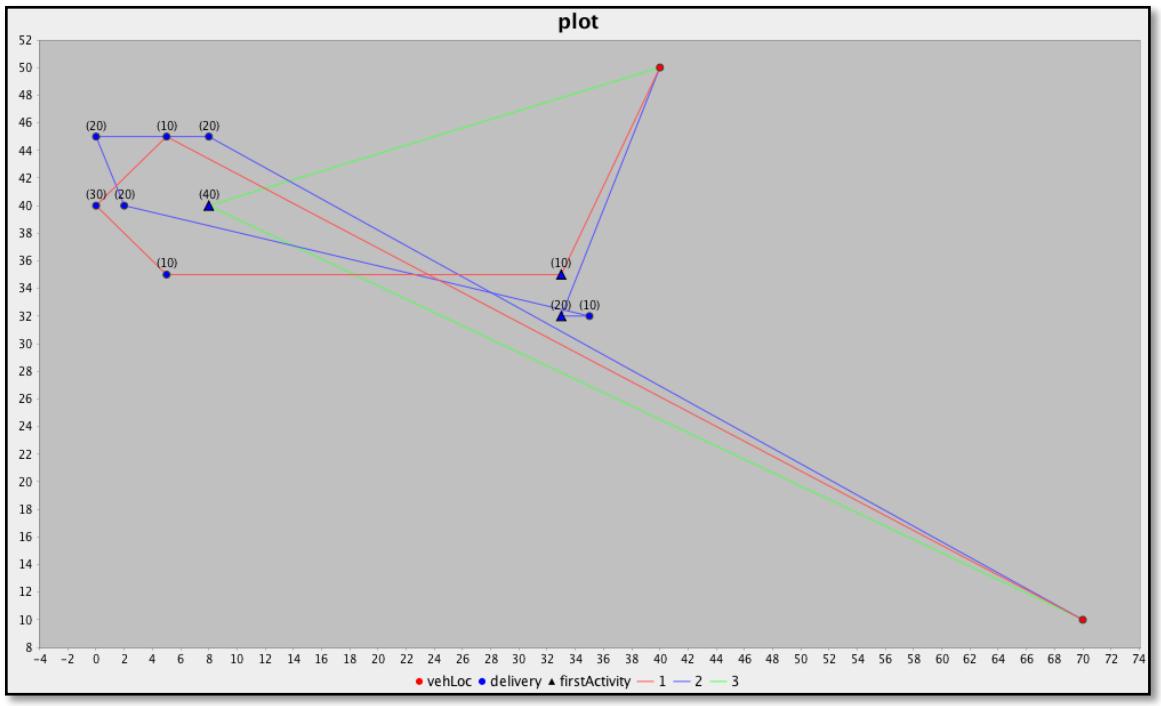

Figure 15: A graph plot of nodes with a 150.0-sec. duration

Detailed Solution							
Route	Vehicle	Activity	Job	ArrTime	EndTime	Costs	Time costs
1	Vehicle	Start	Depot	Undef	0	0	0
1	Vehicle	Delivery	43	17	267	0	17.0
1	Vehicle	Delivery	38	300	729	0	33.0
1	Vehicle	Delivery	34	738	1001	0	9.0
1	Vehicle	End	Destination	1072	Undef	0	71.0
2	Vehicle	Start	Depot	Undef	0	0	0
2	Vehicle	Delivery	42	19	318	0	19.0
2	Vehicle	Delivery	37	350	633	0	32.0
2	Vehicle	Delivery	36	639	915	0	6.0
2	Vehicle	End	Destination	989	Undef	0	74.0
3	Vehicle	Start	Depot	Undef	0	0	0
3	Vehicle	Delivery	41	19	416	0	19.0
3	Vehicle	Delivery	39	453	817	0	37.0
3	Vehicle	End	Destination	895	Undef	0	78.0
4	Vehicle	Start	Depot	Undef	0	0	0
4	Vehicle	Delivery	33	34	337	0	34.0
4	Vehicle	Delivery	35	343	593	0	6.0
4	Vehicle	End	Destination	662	Undef	0	69.0

Table 14: A detailed solution of a 250.0-sec. duration

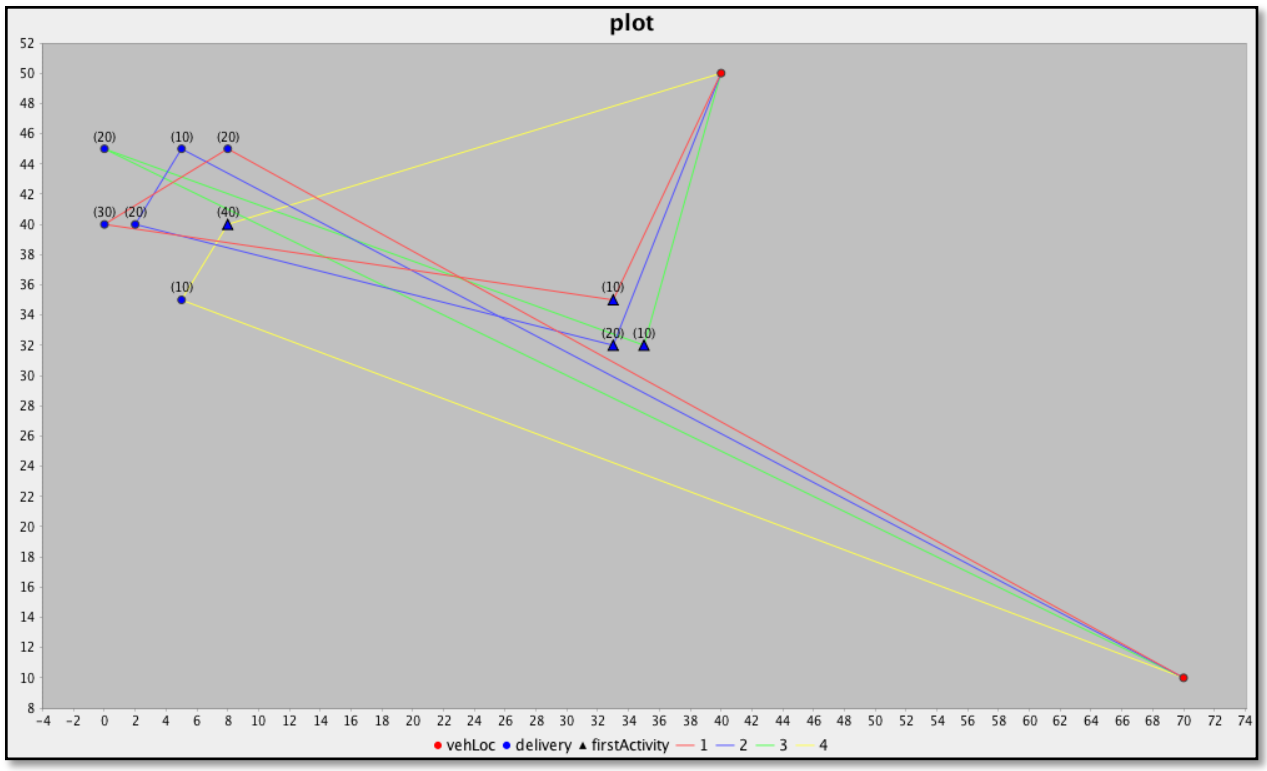

Figure 16: A graph plot of nodes with a 250.0-sec. duration

As a result from the two case studies, the time changes when any of the settings' data changes. From the first case study, which changes the number of nodes, time cost decreases while the number of nodes decreases. In the second use case, which changes the amount of the duration, shows that the shorter the duration of nodes, the fewer agents used in routing. Table 15 sums up the results of both case studies.

	Change number of nodes.			
Experiment #	**Number of nodes**	**Duration**	**Number of decreased agents**	**Time cost without waiting**
1	100	90	14	3571.0
2	50	90	7	1681.0
3	25	90	3	640.0
4	10	90	2	363.0
	Change nodes' duration			
1	10	50	1	196.0
2	10	90	2	247.0
3	10	150	3	375.0
4	10	250	4	504.0

Table 15: Summary of case studies results

CHAPTER SIX: CONCLUSION AND FUTURE WORK

CONCLUSION

The objective of this thesis is to find the optimal time path routed by a minimal number of agents with time windows to serve a number of predefined nodes with time windows. The problem is formulated using Integer Linear Programming. The problem is time constrained, making it difficult to find an optimal solution. A heuristic approach that introduces a solution closer to optimal is developed. The heuristic solution is composed of three stages: first, insertion construction stage to insert the nodes into agents' schedules; then, a combination of A* Search and Ruin and Recreated algorithms to find the shortest time path for each agent; and finally, using the Local Search Method to move a node from one schedule to another under some specifications. In the simulation results chapter, two case studies are introduced to show how changing the number of nodes and nodes' duration effects time cost and the number of vehicles used in routing. The results show that the greater number of nodes available, the more agents used to serve all nodes, as well as an overall increase in routing time. The second case study found that the less service duration, the fewer agents used.

FUTURE WORK

For future work, we plan to remove some of the restrictions and assumptions in order to render the problem more generic to match real-world scenarios. For example taking consider of agent's driver, a vehicle's energy can also limit work and traveling areas. The reason energy is not considered is because energy has an impact on multiple aspects, which makes formulation different and complex. We will introduce more complex

constraints and add real-world conditions to the car and road network, such as vehicles' energy, moving and dynamic road construction areas, and sudden changes in the time windows.

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (IoT): A vision, architectural elements, and future directions," Future Generation Computer Systems, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[2] H. Arasteh, V. Hosseinnezhad, V. Loia, A. Tommasetti, O. Troisi, M. Shafie-Khah, and P. Siano, “Iot-based smart cities: A survey,” 2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC), 2016.

[3] X. Sun, “Study on a dynamic traffic route choice model with travel time reliability constrains of Intelligent Transportation Systems,” 2014 IEEE 5th International Conference on Software Engineering and Service Science, 2014.

[4] G. Dantzig and J. Ramser, “The Truck Dispatching Prob- lem,” *Management Science*, Vol. 6, No. 1, 1959, pp. 80-91. doi:10.1287/mnsc.6.1.80

[5] S. N. Kumar, "A survey on the vehicle routing problem and its variants," Intelligent Information Management, vol. 04, no. 03, pp. 66–74, 2012.

[6] T. Caric and H. Gold, Vehicle routing problem. Rijek, Crotia: InTech, 2008.

[7] S. Belhaiza, "A game Theoretic approach for the real-life multiple-criterion vehicle routing problem with multiple time windows," IEEE Systems Journal, pp. 1–12, 2016.

[8] N. A. El-Sherbeny, "Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods," Journal of King Saud University - Science, vol. 22, no. 3, pp. 123–131, Jul. 2010.

[9] V. S. Kumar, M. R. Thansekhar, R. Saravanan, and S. M. Joe Amali, "Solving multi-objective vehicle routing problem with time windows by FAGA," Procedia Engineering, vol. 97, pp. 2176–2185, 2014.

[10] Z. Xu and J. Tang, "Digital library," The Society of Digital Information and Wireless Communication, 2013, pp. 268–273. [Online]. Available: http://sdiwc.net/digital-library/heuristic-algorithm-of-vrptw-based-on-singlepass-multiple-vehicle-collaboration.html. Accessed: Feb. 10, 2017.

[11] J. L. Gross and J. Yellen, Graph theory and its applications, 2nd ed. Boca Raton, FL: Chapman & Hall/CRC, 2005.

[12] HackerEarth, "Graph representation tutorials & notes | Algorithms," HackerEarth, 2017. [Online]. Available: https://www.hackerearth.com/practice/algorithms/graphs/graph-representation/tutorial/. Accessed: Feb. 10, 2017.

[13] W. Zeng and R. L. Church, "Finding shortest paths on real road networks: The case for A*," International Journal of Geographical Information Science, vol. 23, no. 4, pp. 531–543, Apr. 2009.

[14] M. M. Solomon and J. Desrosiers, "Survey Paper—Time window constrained routing and scheduling problems," Transportation Science, vol. 22, no. 1, pp. 1–13, Feb. 1988.

[15] K. Jansen and S. Öhring, "Approximation Algorithms for time constrained scheduling," Information and Computation, vol. 132, no. 2, pp. 85–108, Feb. 1997.

[16] C. Malandraki and M. S. Daskin, "Time dependent vehicle routing problems: Formulations, properties and Heuristic Algorithms," Transportation Science, vol. 26, no. 3, pp. 185–200, Aug. 1992.

[17] I. H. Osman, "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," Annals of Operations Research, vol. 41, no. 4, pp. 421–451, Dec. 1993.

[18] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews), vol. 28, no. 3, pp. 392–403, 1998.

[19] A. Misevicius, "Genetic algorithm hybridized with ruin and recreate procedure: Application to the quadratic assignment problem," Knowledge-Based Systems, vol. 16, no. 5-6, pp. 261–268, Jul. 2003.

[20] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck, "Record breaking optimization results using the ruin and recreate principle," Journal of Computational Physics, vol. 159, no. 2, pp. 139–171, Apr. 2000.

[21] V. Pureza, R. Morabito, and M. Reimann, "Vehicle routing with multiple deliverymen: Modeling and heuristic approaches for the VRPTW," European Journal of Operational Research, vol. 218, no. 3, pp. 636–647, May 2012.

[22] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "Finding the shortest path in stochastic vehicle routing: A Cardinality Minimization approach," IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 6, pp. 1688–1702, Jun. 2016.

[23] R. Liu, X. Xie, V. Augusto, and C. Rodriguez, "Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care," European Journal of Operational Research, vol. 230, no. 3, pp. 475–486, Nov. 2013.

[24] L. Fu, D. Sun, and L. R. Rilett, "Heuristic shortest path algorithms for transportation applications: State of the art," Computers & Operations Research, vol. 33, no. 11, pp. 3324–3343, Nov. 2006.

[25] graphhopper, "Graphhopper/jsprit: Jsprit is a java based, open source toolkit for solving rich vehicle routing problems," 2017. [Online]. Available: https://github.com/graphhopper/jsprit. Accessed: Feb. 23, 2017.